BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(19)日本国特許庁（ＪＰ）　(12)**公表特許公報（Ａ）**　(11)特許出願公表番号
特表2002－539559
（P2002－539559A）
(43)公表日　平成14年11月19日(2002.11.19)

(51)Int.Cl.[7]	識別記号	ＦＩ		テーマコード（参考）
Ｇ０６Ｆ　13/00	５４０	Ｇ０６Ｆ　13/00	５４０Ｆ	５Ｂ０７５
12/00	５４６	12/00	５４６Ｂ	５Ｂ０８２
17/30	１１０	17/30	１１０Ｆ	
	２１０		２１０Ｄ	
	３８０		３８０Ｃ	

審査請求　未請求　予備審査請求　有　（全 58 頁）　最終頁に続く

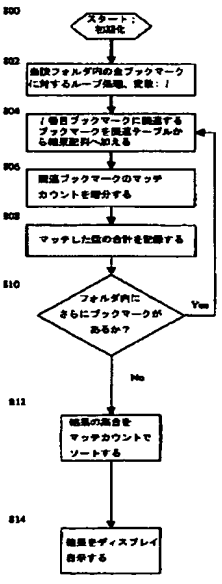
(21)出願番号　特願2000－605902(P2000－605902)
(86)（22)出願日　平成12年２月23日(2000.2.23)
(85)翻訳文提出日　平成13年８月６日(2001.8.6)
(86)国際出願番号　ＰＣＴ／ＵＳ００／０４５８８
(87)国際公開番号　ＷＯ００／５５７４１
(87)国際公開日　平成12年９月21日(2000.9.21)
(31)優先権主張番号　６０／１２５，０４８
(32)優先日　平成11年３月18日(1999.3.18)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　ブリンク　ドット　コム　インコーポレイテッド
アメリカ合衆国、ニューヨーク州　10038、ニューヨーク、フルトン　ストリート　64
(72)発明者　デビッド・シーゲル
アメリカ合衆国、ニューヨーク州、ニューヨーク、イースト　エイティース　ストリート　401、アパートメント　８ジー
(74)代理人　弁理士　藤元　亮輔

Ｆターム(参考)　5B075 KK07 NR06 NR12 PR01 PR03
PR06 PR08 QM08
5B082 PA11 HA05

(54)【発明の名称】　インターネット検索とホットリンクを結びつける、相乗作用を生むインターネットブックマーク

(57)【要約】
様々な側面において、本発明の目的の一つは、インターネット上を往き来しやすくするためにワールドワイドウェブのリンク（ＵＲＬ：ユニバーサル・リソース・ロケータとしても知られている）を管理するとともに自動的にカテゴリ分けして、当該リンクの個人及び公共ディレクトリを作成するシステム及び方法を提供することである。様々な実施形態において、本発明のシステムは、データベースに複数ユーザのウェブリンクを保存しそのウェブリンクを抽出及び表示する方法、本システム中にユーザが保存した既存のリンクに関連のあるウェブリンクを取得する方法等の他関連する特徴を提供する（８０４）。ユーザは、リンクのコレクションを組織化し管理して、他のユーザのリンクコレクションの中から関連するリンクを取得することができる（８０８）。データベース・マッチングの特徴は、関連性を取得するために、年齢、性別、職業等を含むユーザのプロフィール情報など、他の情報を利用することができることである。



【特許請求の範囲】

【請求項１】　複数のユーザからそれぞれウェブリンクのリストを少なくとも一つづつ提供してもらうステップと、

当該リストを分析し、前記複数ユーザのリストの中で密接に関連するペアを異なるユーザの所持するものから組み合わせて一組決定するステップと、

前記関連するペアからのリンクをまとめて一つのリンクリストに統合し、その統合されたリンクリストを、前記複数ユーザ間で共用できるように一つのフォルダに入れて提供するステップとを有するワールドワイドウェブのリンクを管理する方法。

【請求項２】　マスターリンクディレクトリ及びユーザプロファイルを提供するステップと、そのマスターリンクディレクトリを分析して、そのユーザプロファイルに密接に関連する関連コンテンツのリンクのコレクションを形成するステップとをさらに有する請求項１記載の方法。

【請求項３】　他のユーザによるリンクのフォルダへのグルーピングを確認するステップと、他のユーザがそのリンクをどのようにフォルダにグルーピングするかに基づいてそのリンクをカテゴリ分けするステップとをさらに有する請求項１記載の方法。

【請求項４】　ウェブページ上の一セットの見本となるリンクに基づいて、動的リンクコレクションを生成するステップをさらに有する請求項１記載の方法。

【請求項５】　類似したリンクのコレクションを所持するユーザ同士をグルーピングすることによってウェブコミュニティを形成するステップをさらに有する請求項１記載の方法。

【請求項６】　前記関連リンクのフォルダに特定のリンクを挿入するステップをさらに有する請求項１記載の方法。

【請求項７】　見本となるリンク又はリンクコレクションに対してマッチングを行うことによってパブリックリンクフォルダを捜し出すステップをさらに有する請求項１記載の方法。

【請求項８】　前記ユーザがどの程度密接に関連しているかによってマッチ

ングを重みづけするため、あるいは同一の嗜好を有する他のユーザプロファイルにとって関心のあるリンクを表示するために、前記ユーザプロファイルにユーザの嗜好を加えることができる請求項２の方法。

【請求項９】　パブリックフォルダにリンクを追加すると前記ユーザに更新を通知するステップをさらに有する請求項７記載の方法。

【請求項１０】　前記通知ステップには、電子メールによる前記更新の通知を含む請求項９記載の方法。

【請求項１１】　ユーザリンクコレクションから無効リンクを削除するステップをさらに有する請求項１記載の方法。

【請求項１２】　各個人ユーザの前記グルーピング情報をあわせることによって、私的リンクフォルダの集合組織化に基づいてマスターリンクフォルダを生成するステップをさらに有する請求項２記載の方法。

【請求項１３】　前記分析ステップは、リンクフォルダのコレクションを、個人ユーザリンクフォルダの階層を利用してマスターリンクツリーとして最良の表現となる階層を見つけるリンクフォルダの分類法又は階層体系に、組織化することを含む請求項１記載の方法。

【請求項１４】　前記分析ステップは、一ユーザのリンクフォルダを他のユーザと比較することによって、ユーザに階層又はフォルダ組織を提案することをさらに含む請求項１３記載の方法。

【請求項１５】　一般向けの読み取り専用リンクに対して確定した関連ヒット数に基づき、ユーザに報酬を提供するステップをさらに有する請求項１記載の方法。

【請求項１６】　前記ユーザが識別されたマッチングの適切さを評価して、それを、以後のマッチングをさらに精緻なものとするために前記重みづけ機能に組み込むステップをさらに有する請求項８記載の方法。

【請求項１７】　前記複数ユーザの個人ユーザ毎に、及び前記ユーザ全員に対して、リンクに訪れた回数の合計を追跡するステップをさらに有する請求項１記載の方法。

【請求項１８】　（１）特定のリンクのヒストリーであって、当該リンクが

参照された最後の日付、当該リンクが前記複数ユーザのうちの特定ユーザ及び前記複数ユーザ全員によってアクセスされた回数、ユーザによる当該リンクの評価、及び／又は記述的テキスト情報を含むことができるヒストリーと、（２）個々のリンクリストに同じリンクを有する他ユーザから取得した情報とを利用することにより、前記統合されたリンクリストを検索するステップをさらに有する請求項８記載の方法。

【請求項１９】　リンクを後でソートするために未ソートリンクのフォルダを作成することをさらに含む請求項２記載の方法。

【請求項２０】　各未ソートリンクを配置するための前記ユーザリンクコレクション中のもっとも関連性のあるフォルダを提案することによって、前記未ソートリンクフォルダを自動ソートするステップをさらに有する請求項１９記載の方法。

【請求項２１】　特定ユーザと類似したリンクコレクションを所持する他ユーザを確定し、当該他ユーザを類似リンクコレクションを有する全ユーザの動的に作成されるディスカッショングループに参加するよう呼びかけるステップをさらに有する請求項２記載の方法。

【請求項２２】　前記関連リンクのフォルダに特定のリンクを挿入する前記ステップは、

市場開拓すべき製品又はサービスを代表するリンクの見本コレクションを作成し、

ユーザリンクフォルダとマッチングするものを取得し、

特定のリンクを当該ユーザのリンクコレクションに挿入し、

見本とマッチングするフォルダを当該ユーザが開くたびに当該挿入されたリンクを表示することを含む請求項６記載の方法。

【請求項２３】　前記ユーザが当該フォルダを開くたびにページの上部にバナー広告を表示する請求項２２記載の方法。

【請求項２４】　埋め込みリンクリストを作成するステップをさらに有する請求項１記載の方法。

【請求項２５】　遠隔地の複数個人ウェブユーザと、

当該複数ユーザの各々からのウェブリンクリストを少なくとも一つ保存する集中管理されたデータ記憶装置と、

当該複数ユーザのうち異なるユーザからの関連するリストのペアを一組識別し、当該関連するペアからのリンクをまとめて一つのリンクリストに統合する手段と、当該統合したリンクリストをフォルダに入れて当該集中管理されたデータ記憶装置に保存して当該複数ユーザ間で共通にアクセス利用できるようにする手段とを有する中央サーバからなるワールドワイドウェブリンク管理システム。

【請求項２６】　前記集中管理されたデータ記憶装置は前記個人ユーザに対するユーザプロファイルを管理する請求項２５記載のシステム。

【請求項２７】　前記保存されたユーザプロファイルにはユーザの嗜好を含み、前記中央サーバは前記ユーザがどの程度密接に関連しているかによってマッチングを重みづけする、あるいは同一の嗜好を有する他のユーザプロファイルにとって関心のあるリンクを表示する手段を有する、請求項２５記載のシステム。

【請求項２８】　前記中央サーバは、リンクを前記統合されたリンクリストから追加するとき、前記ユーザに更新を通知する手段をさらに有する請求項２５記載のシステム。

【請求項２９】　前記中央サーバは、一般向けの読み取り専用リンクに対して確定した関連ヒット数に基づき、ユーザに報酬を提供することを容易にする手段をさらに有する請求項２５記載のシステム。

【請求項３０】　前記中央サーバは、ユーザが識別されたマッチングの適切さを評価しやすくする手段をさらに有する請求項２５記載のシステム。

【請求項３１】　前記関連するリンクのフォルダに特定のリンクを挿入する手段をさらに有する請求項２５記載のシステム。

【請求項３２】　見本となるリンク又はリンクコレクションに対してマッチングを行うことによってパブリックリンクフォルダを捜し出す手段をさらに有する請求項２５記載のシステム。

【請求項３３】　パブリックリンクフォルダにリンクを追加するとき、前記ユーザに更新を通知する手段をさらに有する請求項３２記載のシステム。

【請求項３４】　前記通知手段には、電子メールによる通知を含む請求項３

３記載のシステム。

【請求項３５】　一般向けの読み取り専用リンクに対して確定した関連ヒット数に基づき、ユーザに報酬を提供する手段をさらに有する請求項２５記載のシステム。

【請求項３６】　前記ユーザが識別されたマッチングの適切さを評価することを可能にし、当該評価を、以後のマッチングをさらに精緻なものとするために前記重みづけ機能に組み込む手段をさらに有する請求項２７記載のシステム。

【請求項３７】　前記複数ユーザのうちの各個人ユーザ及び前記ユーザ全員に対して、リンクに訪れた回数の合計を追跡する手段をさらに有する請求項２５記載のシステム。

【請求項３８】　（１）特定のリンクのヒストリーであって、当該リンクが参照された最後の日付、当該リンクが前記複数ユーザのうちの特定ユーザ及び前記複数ユーザ全員によってアクセスされた回数、ユーザによる当該リンクの評価、及び／又は記述的テキスト情報を含むことができるヒストリー、及び（２）個々のリンクリストに同じリンクを有する他ユーザから取得した情報を利用することにより、前記統合されたリンクリストを検索する手段をさらに有する請求項２７記載のシステム。

【請求項３９】　マスターリンクディレクトリ及びユーザプロファイルを提供し、当該ユーザプロファイルに密接に関連する関連コンテンツのリンクのコレクションを形成するために当該マスターリンクディレクトリを分析する手段と、

リンクを後でソートするために未ソートリンクのフォルダを作成する手段と、

各未ソートリンクを配置するための前記ユーザリンクコレクション中のもっとも関連性のあるフォルダを提案することによって、前記未ソートリンクを自動ソートする手段とを有する請求項２５記載のシステム。

【請求項４０】　前記関連リンクのフォルダに特定のリンクを挿入する前記手段は、

市場開拓すべき製品又はサービスの代表的リンクの見本コレクションを作成する手段と、

ユーザリンクフォルダとマッチングするものを取得する手段と、

特定のリンクを当該ユーザのリンクコレクションに挿入し、

見本とマッチングするフォルダを当該ユーザが開くたびに当該挿入されたリンクを表示する手段とを含む請求項３１記載のシステム。

【請求項４１】　埋め込みリンクリストを作成する手段をさらに有する請求項２５記載のシステム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

関連出願への相互参照

本出願は、アメリカ合衆国仮特許出願シリアル番号60/125,048号に関連し、その優先権の利益を主張するものである。

【０００２】

発明の属する技術分野

本発明は、一般にインターネット情報のアクセスに係り、特にウェブページへのアクセスの自動管理を楽にすることに関する。

【０００３】

著作権に関する許諾

本特許書類（図面及び補遺を含むがそれらに限定されない）の開示には、その一部に著作権保護を受ける素材を含んでいる。その著作権者は、この特許書類及びアメリカ合衆国特許商標庁で維持される関連ファイルないし記録の謄本を複製することに異議なしとする。しかしながら、これによって、この著作権者はそれ以外に係るすべての著作権を留保するものである。

【０００４】

【従来の技術】

背景

インターネットのワールドワイドウェブ（あるいは単に「ウェブ」）は人々が情報を保存、配布する場所である。ウェブの有用性は、人々が自分の関心事に関連する題材を捜し出すことができるかどうかにかかっている。ウェブの成熟につれ、ウェブ上で利用可能な情報の量が劇的に増加した結果、特に興味のある題材に関連する情報だけを得ることがますます困難になってきている。

【０００５】

ウェブとは、本質的に、蜘蛛の巣（spider's web）に似たものとしてイメージすることができるようなリンクされたコンテンツページのコレクションである。ある場所に保存された情報を別の場所にいる誰かがいかに容易に捜し出すことが

できるかによって、ウェブの有用性はその大方が決定する。ウェブページに保存される情報は、典型的に、ウェブブラウザ（例えば、ネットスケープやインターネットエクスプローラ）と呼ばれるプログラムを通じて、「ユニバーサル・リソース・ロケータ」すなわち「ＵＲＬ」を介してアクセスされる。ＵＲＬは、一般に「ウェブアドレス」、「ハイパーリンク」あるいは単に「リンク」と言い習わされている。ＵＲＬは、例えば http://www.ibm.com/ のように記されるものである。ローカルコンピュータ上のウェブブラウザにＵＲＬを入力すると、ブラウザはウェブサーバに接続し当該ＵＲＬに結合した特定のページのコンテンツを表示する。ウェブページにはさらにハイパーリンクを埋め込んでおくことが可能であり、それを何らかの方法（例えばマウスでクリックする等）で選択すれば、同様の仕方でブラウザは対応するページに移動する。

【０００６】

インターネットが進化するにしたがって、ハイパーリンクが有用となるような関連コンテンツ取得方法も進化してきた。現在の主なリンク又はウェブディレクトリの管理方法で、今日利用されているものには次のようなものがある：

・ユーザが自分のコンピュータにローカル保存できてブラウザでアクセスすることができるリンクリスト

・ユーザが訪れたサイトの日付と時間で組織化したリストを含むローカル保存されたヒストリーファイル

・手作業で集めてウェブページ上に配置したリンクリスト（例えば、団体のホームページ）

・関連するコンテンツを有するページ同士をリンクするウェブリンクの輪

・すべてのウェブリンクの一覧表を作り、ユーザが特定のコンテンツを検索できるようにしたサーチエンジン

【０００７】

ウェブページには、リンクを（たいていは様々な主題毎に）集めたリストを含んでいることが多い。ウェブ上でいくつかの場所を往き来するには、一般に、ブラウザにＵＲＬをタイプするか、一つないし複数のリンクのリストからＵＲＬを選択することによるので、素材の所在は典型的にハイパーリンクのリストとして

ユーザに提示される。このようなリンクを通常「ホットリスト」と呼ぶ。

【０００８】

【発明が解決しようとする課題】

アルタヴィスタ（AltaVista）やインフォシーク（Infoseek）のように、自動的、継続的にウェブの全コンテンツを調べ上げてアドレスとキーワードのデータベースを構築する自動ウェブ「サーチエンジン」も一般的である。ユーザは、アルタヴィスタのデータベースをサーチし、関連性を有する可能性があるコンテンツを探すことができる。これらの自動ウェブサーチエンジンはユーザがクエリで与える一組のキーワードに対してテキストのマッチングを行う。単にそのテキストを含むページを取得するだけであって、必ずしも求める概念に関連するページを取得するわけではない。その結果、困ったことに、サイトへのアクセス数を稼ごうとして、多くのウェブページ作者がサーチエンジンを欺き、ユーザが何か別のテーマで一般的なサーチを実行したときのサーチ結果として自分のページを返すよう企てるようになる。サーチエンジンのユーザのほとんどがかかる問題を経験している。例えば、書店のウェブサイトが、ホワイトハウス（the White House ：大統領官邸）のホームページを取得しようとして誰かがサーチを実行した際に自分のリンクが表示されることを期待して「ホワイトハウス（white house ：白い家）」のような言葉をホームページ中に挿入するかもしれない。

【０００９】

ヤフー（Yahoo）やマイニングカンパニー（Mining Company）のようなサイトは、組織化されたリンクの膨大なコレクションを維持しているが、これは、手作業で集めてカテゴリ毎にグルーピングすることでユーザの利用に供するものである。

【００１０】

最後に、ほとんどのブラウザは、ユーザがお気に入りのウェブページのアドレスを手作業で保存することを可能にするメカニズムを有する。この特徴は、一般に「ブックマーク」又は「お気に入り」と呼ばれている。典型的に、ユーザがウェブを往き来するうちに興味のあるコンテンツを含むページを取得すると、それを「ブックマーク」し、自分の「お気に入り」に追加して、再度検索する手間抜

きに簡単にそのページに戻ってくることができる。このことから、すべてのブックマークを一つの未組織化状態のグルーピングのまま放置することで、組織化されていない使い難い個人ブックマークのコレクションができあがってしまうことも多い。しばらく後の時点で、ユーザはブックマークをソートし、関連するフォルダ毎にそれを配置する。例えば、インターネットエクスプローラ又はネットスケープのブックマーク機能を利用すると、ユーザは、自らがすでに作成したフォルダの（膨大な量になる可能性もある）コレクションの中に新しいリンクをファイルするために適当なフォルダを探さなければならない。関連するフォルダが見つからなければ、ユーザは、その新しいリンクを保存するための新しいサブフォルダをその中に作成する既存のフォルダを決定しなければならない。ユーザが制作するブックマークのコレクションが膨大になるほどに、新しい登録項目を組織化してリンクを保存する最良の場所を決定するのにより一層多くの時間がかかるだろう。

【００１１】

【課題を解決するための手段】

要約及び発明の目的

様々な側面において、本発明の目的の一つは、インターネット上を往き来しやすくするためにワールドワイドウェブのリンク（ＵＲＬ：ユニバーサル・リソース・ロケータとしても知られている）を管理するとともに自動的にカテゴリ分けして、当該リンクの個人及び公共ディレクトリを作成するシステム及び方法を提供することである。様々な実施形態において、本発明のシステムは、データベースに複数ユーザのウェブリンクを保存しそのウェブリンクを抽出及び表示する方法、本システム中にユーザが保存した既存のリンクに関連のあるウェブリンクを取得する方法等の他関連する特徴を提供する。

【００１２】

本発明のシステム及び方法により、ユーザは、リンクのコレクションを組織化し管理して、他のユーザのリンクコレクションの中から関連するリンクを取得することができる。データベース・マッチングの特徴は、関連性を取得するために、年齢、性別、職業等を含むユーザのプロフィール情報など、他の情報を利用す

ることができることである。本質的には、本システムの全参加ユーザがウェブディレクトリを組織化することに貢献する。

【００１３】

本発明は、システムの全ユーザの収集・組織化努力に基づき、自己組織化して関連サイトのクラスターとなるリンクのリストを生成する。従来システムと異なり、キーワードやフルテキスト検索に基づくものではない。ユーザがすでに取得した既存コンテンツに関連するコンテンツを取得することができる検索・クラスター化メカニズムを作成するために、個人の関連ウェブリンクのクラスター化操作において含意された情報を利用する。本発明によって、利用しやすさがもたらされ、ユーザが明示的にカテゴリを決定する必要はなくなる。このアプローチは、全参加ユーザの経験を活用する自己組織化するウェブディレクトリを作成する。

【００１４】

本発明の自己組織化するリンクリストによって、ワールドワイドウェブのユーザが興味深いと感じるコンテンツへの自らのウェブリンクを組織化するとともに現存するサーチエンジンで興味があり関連があるコンテンツを捜し出すことが可能であり、より効率的なリンクの管理が可能であり、従来のキーワードに基づくサーチエンジンのもたらす結果よりも具体的でマッチングの誤りも出にくく、手作業の介在なしで形成され、ユーザの実際の嗜好と使用形態を反映しカタログサーチサイトの編集者である人間の編集上の偏りや知識の制約から比較的自由であって、興味深い関連コンテンツを取得した他のユーザの収集努力を反映しその成果を共有するシステマチックなアプローチ方法が提供される。

【００１５】

本発明の前記その他の特徴及び利点は、添付図面に示されているような、以下に記述するその例示的実施態様の詳細な説明を参酌することによりさらに明らかとなるであろう。

【００１６】

【発明の実施の形態】

詳細な説明

図１に示すアーキテクチャの好適な実施態様において、各々の個人ウェブユーザ１、２、・・・ｎ（例えば、各ユーザはＰＣを介してインターネットにアクセスできる）は、中央サーバコンピュータ１０と、好ましくはローカルウェブブラウザ（例えば、ネットスケープやインターネットエクスプローラ）によって、セッションを開始する。中央サイト１００において、中央サーバコンピュータ１０はデータベースが置かれたデータ記憶装置２０にアクセスする。この好適な実施態様では、ユーザは公共のインターネットを通じてＰＣによって本発明にアクセスする。しかし、代替的に、本発明の実施が私的なイントラネットで行われる実施態様もあり得るし、ウェブユーザアクセスの代替技術、例えば、携帯電話によるアクセスやウェブＴＶ（Web TV™）等を利用する実施態様も可能である。最適なパフォーマンスを達成するために、システムは、代替的諸実施態様において、２以上の中央コンピュータ及び２以上のデータベースで実行することができる。データベースの複製を継続的に同期させ、複数サーバ間のウェブトラフィックの負荷を分散させるため、従来のソフトウェア的解決方法が利用される。

【００１７】

本発明において、ウェブブラウザの「お気に入り」又は「ブックマーク」の機能は、単独共通サイト１００で集中管理される。本サービスの全ユーザが自身のブラウザの分散（ローカル）ブックマーク機能を利用せずに共通中央サイト１００に各自のブックマークを保存する。より詳しい検討は後述するが、個人のユーザが作成し中央サイトのデータベース２０に保存されたブックマークのコレクション又はリストは、定期的にリストを対にして比較することによって分析する。２つのリストの関連が深いことが分かったら、その２つのリンクは関連ありと見なされ、新規の集中管理されるリストが、元の２つのリストを完全に又は部分的に結合してランク付けされたものとして生成される。すると、本システムの他のユーザが、その元になった２つのリストの所有者と同様、この新規の集中管理されたリストにアクセスしそれを利用することになる。元のリストのいずれか一方のみよりも包括的な関連サイトのリストが提供されるのである。

【００１８】

この処理によって、インターネットのユーザは、ＵＲＬのブックマーク（リン

ク）を検索し、グルーピングして、中央サーバ上に保存されインターネット上のどのウェブブラウザからでもアクセスできる階層構造を有するフォルダ群にまとめることができる。個人ユーザ・フォルダのグルーピングをまとめた結果に基づきリンク間の関係が自動取得されることで、ユーザはインターネット上の関連情報を捜し出すことができる。例えば、この処理は、ユーザが以前ひとつのフォルダにまとめたリンクと類似したリンクをそのユーザに提案してくれる。他のリンクは、同様の方法でグルーピングした他のユーザの数や、ユーザフォルダのリンクコンテンツを横断することによってリンクを関連づける距離メトリックや、ユーザによる評価や、ユーザの経歴に関するプロファイル情報による等いくつかの基準に基づく類似性の尺度を基礎に特定し順位づけて並べたリンクである。

【００１９】

ウェブユーザは、指定したウェブサイトに接続することによって、本発明にアクセスする。好適な実施態様による主システムフロー経路の例を図２に示す。ユーザログイン２００では、ユーザ名とパスワードをウェブベースのアプリケーションから直接システムにクエリで入力するか、「クッキー（Cookie）」等のブラウザ側で提供する機能により渡し、フローを開始する。システムは、２０２～２０４で、ユーザアカウントのデータベースを参照しこの情報の正当性を認証する。ユーザが登録ユーザでなければ、２０３で登録手続きを開始してユーザ登録し、２１３でユーザアカウントを作成し、フローは２０６に続く。無効なユーザ名／パスワードの組み合わせが入力されると、２０５でエラーメッセージが発行され、ユーザにユーザログイン２００に戻ってやり直すよう促す。ユーザ名とパスワードがデータベースの有効登録と一致すると、システムは、２０６で、当該ユーザのホームページでそのユーザのための新しいウェブセッションを開始し、２０８でトップレベルリンク及び当該ユーザに関連してセットされたディレクトリをリストアップするカスタムホームページディレクトリを生成（又は表示）する。この実施態様（添付の図３参照）では、表示されたページ（及びそれに続くページ）は、リンク３０（添付の図３参照）又はユーザが様々なリンク関連機能を実行することができるボタン３２（添付の図３参照）を含む。例えば、ユーザはリンク（例えばブックマーク）２１０をクリックして対応するウェブページ２１

１に行くとともにそのサイトに対応するヒット数２１２を更新することができる。また、ユーザはフォルダ２１４をクリックして対応するフォルダコンテンツ２１５（さらに多くのリンク及びフォルダ等を含む）を表示することができる。また、ユーザは代替的に所定のコマンド２１６セットをクリックし、例えば、サブディレクトリ（もしあれば）に降り、ディレクトリ階層の前のディレクトリに戻り、現在のディレクトリに新規リンクを追加し、現在のディレクトリからリンクを削除する等の、対応する操作を行う２１７ことができる。

【００２０】

このようなリンクの組織は、ファイルとフォルダの階層構造に組織化された標準的なコンピュータのファイルシステムに類似する。多くのウェブベースのディレクトリ、例えばヤフーで用いられているアプローチに類似しているが、ヤフーのようなサイトには、万人に共通のウェブディレクトリが一つ存在する。ヤフーのユーザが往き来することができるのは、唯一そのディレクトリだけである。反対に、本発明のシステムでユーザがリンクをクリックすると、システムはそのユーザのデータベース中の選択されたリンクの入力項目に日付と時間を記録し、そのユーザがそのリンクをクリックした回数の総数を追跡し続けるためにカウンタを増分する。そして、その同じ情報がそのリンクに対するプールされた全ユーザの情報を保持する総計データベースに追加される。したがって、システムは、リンクへの訪問回数の総数を、各個人ユーザ毎に、及び全ユーザに対しても、追跡し続けていることになる。次に、システムは、選択されたページにウェブコントロールを転送し、ウェブブラウザはその選択されたページを（あるいは新しいウィンドウに）表示する。

【００２１】

本発明の上記実施態様による画面レイアウトの例を図３に示す。主画面は、リンク３０及び図２を参照して説明した様々な機能を実行するボタン３２を各種提供しているが、これらのリンク及びボタンをクリックすると、例えば、リンクに飛んだり新しいリンクとフォルダを追加したりすることができる。「追加（Add）」ボタンをクリックしてアクセスする「新しいリンクを追加」画面には、当該リンクの名前や説明や評価を入力するフィールドがある。「新しいフォルダを追

加」のフィールドによって、ユーザはフォルダの名前や説明を指定し、そのフォルダが「パブリック」であるかどうかを指示することができる。パブリックフォルダは、他のユーザが参照・閲覧可能で、リンク共有の方法を提供する。

【００２２】

本システムは、各パブリックフォルダを参照したユーザ数を追跡し、そのフォルダの人気等を決定する。利用を容易ならしめるためブラウザのオプションとなっている「アドオン」によって、現在表示しているページを自動挿入することもできる。新規リンクが追加されると、システムは、以前他のユーザでそのリンクを追加した者がいるかどうかチェックする。いなければ、システムは「データベーステーブル」にそのリンクの新しい項目を作成し、固有の「リンクＩＤ」をそのリンクに割り当てる。そして、システムは、このリンクの項目を表示されている、現在のユーザリンクディレクトリに挿入する。内部的には、このリンクは、テキストのＵＲＬではなく、この「リンクＩＤ」によって参照される。それによって、システムは、リンクを参照したすべてのユーザを容易に特定することができるとともに、後述する他の特徴も可能となる。ひとたびリンクが追加されると、そのヒットカウント及び最近アクセスの日付及び時間がそれぞれ「０」と「なし」に初期設定される。そのリンクがその後クリックされると、上述のように、前記フィールドの更新を生成する。

【００２３】

図４Ａはユーザを追跡するためのユーザアカウントデータベーステーブルの例を示す。本システムのユーザには、そのユーザに関する説明情報を含む固有のアカウントが付与される。その情報はデータベースに保存される。ユーザ情報のいくつかのカテゴリを図４Ａの「ユーザ定義テーブル」に示してあるが、そのユーザを所望の分類に割り振るのに役立つよう、他の情報をこのテーブルに動的に追加することもできる。図４Ｂの「フォルダマッピングテーブル」に示すようなデータベーステーブルがユーザのためにフォルダをマッピングする。このような、ユーザ及びフォルダの情報は、後述する検索やマッチングの処理で利用される。

【００２４】

アクセス情報及びリンクはユーザ毎にデータベースに保存されているので、ユ

ーザが自身のリンクコレクション間を往き来するのを容易にする様々な機能を実行することができる。そのような機能の一つに、ユーザが作成したフォルダ内のリンクを取得する検索機能がある。この検索機能は、当該リンクが参照された最後の日付、当該ユーザ及びシステムの全ユーザがそのリンクにアクセスした回数、ユーザによる評価、及び記述的テキスト情報を含むデータのヒストリーを活用する。また、この検索機能は、同じリンクを自分のコレクションに追加した他のユーザから取得した情報も利用する。例えば、システムの全ユーザを通じて最高総合ヒット数を有する該ユーザのコレクションからのリンクを特定するため、検索を実行することができる。これによって、システム全体の中で該ユーザのコレクション中もっともよく利用されたリンクが特定されることになり、そのリンクが一般的な関心が高いことを示唆することとなる。

【００２５】

本発明の重要な特徴の一つは、類似するリンクをグループにまとめる自動カテゴリ分け技術である。リンクは各ユーザが作成する任意の名称を付したフォルダにまとめられる。（フォルダにはサブフォルダを、さらにサブフォルダ内にもサブフォルダを、含むことができる。）自動カテゴリ分けは、ユーザがリンクのセットを一つのフォルダにまとめたときに暗黙の裡に情報を集めてそれを活用することによって行われる。自動カテゴリ分け技術によれば、ユーザによるカテゴリ分けの決定の集合体を利用することで、本質的にこれによらなければ取得することが困難ないし不可能なインターネット上の情報を捜し出すことが容易になる。このカテゴリ分け処理は、特定のフォルダにリンクを配置することのみに依存する。リンクに対し、ユーザが自分でカテゴリ分けを指定するわけではない。

【００２６】

フォルダグルーピング中の暗黙のカテゴリ分け情報を利用することにより、ユーザが手作業でリンクにカテゴリを割り当てることに基づく従来のアプローチのいくつかの主な欠点が克服される。そのような問題の一つは、同じカテゴリなのにユーザによって選択する名称が異なることである。オンライン書店に対するリンクに、ある人は「ショッピング（shopping）」カテゴリを、別の人は「ブックストアズ（bookstores）」、また別の人は「ブックストア（bookstore）」を割

り当ててしまうのである。非英語圏のユーザが自分の母語でカテゴリの名称を選択することもあり得る。こういった問題すべてが、カテゴリの相互関連づけを困難にする。さらに、ユーザにリンクへのカテゴリ割り当てをさせるためには、ユーザの多くが実行することを望まない時間のかかる処理が必要とされる。ユーザがリンクを配置したフォルダに基づく暗黙カテゴリ分けアプローチは、この問題を克服する。

【００２７】

フォルダマトリックス（図５に示す）は、特定のフォルダに保存されたリンクのコレクションを示す。図５において、フォルダはテーブルの縦列で表しＡ～Ｆの番号で表示されている。本システムのリンクは横列で表し１～７の番号で表示されている。マトリックスのセル中に「Ｘ」とあるのは、チェックした縦列のフォルダにチェックした横列のリンクが含まれていることを示す。図５のフォルダマトリックス例では、セルＡ１及びＡ２にチェックマークがみえることからわかるように、フォルダ「Ａ」にはリンク「１」及び「２」が含まれている。テーブルのリンク番号は、そのリンクの追加情報が保存されているリンクデータベース内にある固有のリンクＩＤを示している。図５の「フォルダマトリックス」のフォルダは本システムの一人以上のユーザによって作成及び維持されるフォルダを示している。換言すれば、フォルダ「Ａ」～「Ｆ」は必ずしも特定の一ユーザのものだとは限らない。これらのフォルダの所有者情報は、データベーステーブルに保存されている。便宜的に、Ａ１はフォルダＡ内のリンク１を示すものとする。

【００２８】

このように、リンクとフォルダのデータベースは、ユーザ毎に収集・管理される。すなわち、（固有のユーザアカウントＩＤによって特定される）システムのユーザがそれぞれそのようなフォルダ及びリンクのコレクションを有する。システムに保存されたウェブリンクをカテゴリ分けするためにこれらのコレクションを全面的に利用するアプローチは、総体として、フォルダの類似性の評価に基づくものである。類似するフォルダとは、類似するリンクコレクションを有するフォルダと定義づけられる。例えば、多くのユーザが菜園に関するリンクを収集し

ている場合、そのようなユーザが収集した菜園リンクには重複するものがあると予想される。菜園リンクコレクションのすべてが同じわけではない。あるユーザは自分以外のすべてではないとしてもほとんどのユーザのフォルダ中にはないリンクを持っている。システムは、リンクフォルダを検索して見本フォルダがヒットすると、その見本フォルダ中のリンクと関連がありそうな他のリンクを取得する。

【００２９】

異なる本システムのユーザがリンクを保存したフォルダは、リンク間の関係についての情報を提供する。例えば、リンク１がフォルダＡ及びＢの両方でリンク２と一緒に現れるため、リンク１がリンク２に関連づけられる。フォルダＡ及びＢを作成した本システムのユーザはリンク１と２を同じフォルダ内にまとめて組織化するのが望ましいことに気づいた。リンク１と２はフォルダＡ及びＢ内に一緒に現れるので、システムはそれらを互いに直接的な関連があるものとみなす。リンクはフォルダを介して互いに間接的に関連していることもある。リンク１とリンク４を例にとる。この場合、リンク１とリンク４の両方を含むフォルダは（図５のテーブルにこれらのリンクの両方に対し「Ｘ」がある縦列はないので）特に存在しない。しかし、リンク１と４は、リンク４を含むフォルダＤのリンク３を介して間接的に関連している。フォルダＤはリンク３も含む。リンク３はフォルダＢを介して直接的にリンク１に関連づけられている。したがって、リンク１と４はリンク３を介して互いに関連している。フォルダＢの作成者は、リンク１と３を、その間の関連性を獲得するために一緒にしたほうがよいと考えた。したがって、リンク１、３及び４の間の間接的な関連性はあきらかである。本システムの異なるユーザがフォルダを作成しリンクをその中に入れるため、ユーザ側でリンクをフォルダに組織化するステップ以上に明示的にカテゴリ分けする努力をしなくてもリンク１と４の間の関連性が獲得された。上述の処理は、ユーザがそれを関連フォルダに入れたという事実以上の関連性の性質に関する情報を特に有するわけではない。

【００３０】

上記の例に戻ると、ユーザは、菜園に関連するコンテンツを有するサイトへの

リンクをフォルダに入れておくことができる。フォルダ内には、にんじんに関するウェブページと、トマトに関するものの、二つのリンクがある。別のユーザはにんじんとエンドウ豆に関するリンクのコレクションをフォルダの中に入れている。第三のユーザはエンドウ豆と胡椒のリンクのコレクションを有している。このフォルダコレクションには、胡椒のリンクとにんじんのリンクを両方含む単独のフォルダは存在しないので、胡椒のリンクをにんじんのリンクに関連づけることができる個別のフォルダはない。一方、フォルダの一つの中でにんじんとエンドウ豆が結びついているので、にんじんと胡椒との間のパスをたどることはできる。このような間接的な結びつきは、リンクがすべて一つのフォルダ内にある場合の直接的な関連ほど強くはない。したがって、異なるフォルダにあるリンクどうしの関連性の採点は同一フォルダ内のリンクの評価点より小さくなる。

【００３１】

このカテゴリ化処理には多くの用途がある。重要な用途の一つは、ユーザがすでに収集しフォルダに保存したリンクに関連のあるリンクをさらに取得できるようにすることである。前の例のユーザは、「園芸」に関するリンクのセットを含むフォルダを作成したかもしれず、あるいは、そのフォルダには「菜園」などのようなより具体的なリンクを含んでいたかもしれない。そのユーザは、検索を開始するボタンをクリックし、関連・関係のあるリンクのリストを表示する。システムの全リンクフォルダを検索して情報を収集し、ユーザの「園芸」フォルダ（「見本フォルダ」）と類似するようにみえるものを取得し、最もよく一致したものを提示する。ユーザの見本フォルダに最もよく一致する「パブリックフォルダ」を取得するために同様の検索アプローチが使用される。同様に、ユーザはボタンをクリックし、システムは最も近いようにみえる（が、ユーザのフォルダ中にあるリンクの全体集合を含む）ものを取得するためにすべてのパブリックフォルダを検索する。マッチング処理の詳細について以下説明する。自分のフォルダツリーに当該パブリックフォルダを結びつけたユーザが他に何人いたかによって、そのパブリックフォルダの人気度順にマッチング結果を並べる。ユーザが特定のパブリックフォルダリンクコレクションが有用であることに気づくと、それを読み取り専用フォルダとして自身のリンクコレクションに追加するための適当なボ

タンをクリックする。そのフォルダの所有者がこのフォルダに対して施した変更は即座に全読み取り専用コピーに反映する。また、そのパブリックフォルダリンクコレクションの所有者が追加や変更を行うたびに電子メールのメッセージを送信する特徴を、ユーザは利用可能にすることができる。前記「園芸」の例では、「園芸」に関する有用なリンクのコレクションを有するパブリックフォルダが見つかると、ユーザはそれを自身のフォルダツリーに追加し電子メールの変更通知を利用可能にすることができる。所有者が園芸に関する新規リンクを追加するたびに、このフォルダに接続した人はだれでも自動的にその追加を知らせる電子メール通知を受信する。

【００３２】

上述した従来のブックマーク及びお気に入り機能の利用とは対照的に、この処理を促進するために本システムの自動カテゴリ化機能を利用することによって、リンクを組織化されたものに保つ作業が容易になる。ユーザは「未ソート」リンクの特別なフォルダにブックマークしたリンクを収集しておいて後で組織化する。ユーザが自身の「未ソート」フォルダを組織化する準備が整うと「ソート」ボタンが押され、システムは、自動的に、ユーザの個人リンクコレクション内の各未ソートリンクを配置する最良のフォルダを提案する。これを遂行するために、本発明は、他のユーザにそのリンクをフォルダに配したものがいるかを確認するためグローバルデータベースを検索する。マッチングするものがあれば、そのリンクが配置されたフォルダを、後述するフォルダマッチング・カテゴリ化処理を用いて、ユーザの個人ディレクトリ内の全フォルダコレクションと比較する。そして、そのリンクを保存する最良の場所としての最良マッチングフォルダが提案される。ソートすべきリンクが、グローバルディレクトリ内の二つ以上のフォルダに見つかった場合には、各フォルダをユーザフォルダコレクションに対するマッチングにかけて、総合的に提案されるファイルの場所が計算される。

【００３３】

採点機能は、リンク間の関係の強度を採点する。採点テーブルを図６に示す。図６は、フォルダＡ中のリンク１と他のリンク（２～７）の関係に対する採点処理を表す。因子として使用されるものとしては、関係の直接度（あるリンクを別

のリンクに結びつけるのに要するフォルダの数）、関係を獲得するフォルダの数、及び当該関係パスにおけるリンクの利用頻度などがある。点数は、リンク間の最短フォルダパスの長さの逆数として計算される。したがって、パスが２であれば、得点は、１／２＝０．５である。リンク間にパスがなければ得点は０である。

【００３４】

図５の「フォルダマトリックス」はＡ及びＢの両フォルダにおいてリンク１及び２が直接的関係を有する例である。システムには、２つの異なるフォルダがあり、リンク１及び２が一緒のグループにまとめてある。システムにこれら２つのリンクを有するフォルダが多ければ多いほどリンク１及び２の関係に割り当てられる得点は高くなる。代替的に、リンク１及び２を一つのフォルダにまとめたユーザが少なければ、リンク同士に意味ある関係が本当にあることは不確かであり、得点は低くなる。

【００３５】

フォルダＢ及びＤ内のリンク３を介してリンク１及び４の間等に認められる間接的関係度を評価するためにも同様の採点機能が利用される。１及び４の間に間接的なリンクが多ければ多いほど本当の関係が存在する可能性は高いので、高得点がつく。さらに、直接度が採点機能の因子となる。２つのリンクを関連づけるために横断する必要がある間接的リンク／フォルダの交差が多ければ多いほど、得点は小さい値となる。採点機能はパスの長さの関数として得点を低くする。線形減少や指数関数的減少を用いるなど異なる公知のアプローチを使用できる。指数関数アプローチは得点の減少をパスの長さが線形的に増加するにしたがって加速するものである。採点機能に利用される別の因子として、ユーザが頻繁にアクセスしなかったり最近アクセスしていなかったりするリンク間のパスを割り引いて、頻繁に又は最近利用されたリンクを有するパスを重く評価することがある。例えば、リンク１及び４の結びつきはフォルダＢ及びＤ内のリンク３を介したものであること（図５「フォルダマトリックス」参照）を考慮せよ。この結びつきの得点は、フォルダＢ及びＤの所有者が最近又は頻繁にはリンク３にアクセスしていなければ減点される。

【００３６】

ユーザプロファイル（図４Ａ参照）の類似性も採点機能で使用される。プロファイル相関機能は、二つのユーザプロファイルの関連度を測る尺度を提供してくれる。例えば、米国在住で年齢がそれぞれ４２歳と４９歳の医師であることを示すプロファイル情報を有する二人のユーザは統計的に４２歳の米国の医者と２０歳の中国出身の農夫よりも、関連性のあるリンクコレクションを所持する可能性が高い。したがって、より関連性の高いユーザのフォルダ内に共通にあるリンクは、関連するとは思われないユーザのフォルダ内の同じリンクより高得点がつく。このアプローチの結果、特定のユーザのプロファイルにあわせてカスタマイズされたマッチングが提案される。システムによって、ユーザは、他者がどのようにリンクをグループ分けしているかを探索するため、代替プロファイル情報を代わりに利用することができる。関節炎の処置に関するリンクのコレクションを有する若手医師は、年長者が収集した関節炎リンクコレクションを欲する。システムはこの医師が自分のプロファイルを対象ユーザのプロファイルに置き換えて、その対象プロファイルに与えられる最もよく一致するリンクを取得することを可能にする。

【００３７】

図６から、特定のリンクがフォルダＡ中のリンクにどの程度密接に関連しているかについて判断される。この例では、リンク３が、最も密接に関連しており、３という高得点を示している。リンク４，５，６，及び７の関連性は漸進的に得点が低減しているので、関連性が漸進的に減少している（図５「フォルダマトリックス」参照）。ここで、リンク３はフォルダＢ中に現れており、フォルダＢにはフォルダＡ中の全リンクも含まれる。したがって、フォルダＢはフォルダＡの全リンクとリンク３を含むことになる。これは、リンク３とフォルダＡのリンクとの間に密接な関連性がある可能性を示唆する。システム中にフォルダＡのリンクとリンク３を含むフォルダがもっとあれば、リンク３の得点はそのようなマッチングの数に比例して増加する。別の例をあげると、リンク７はフォルダＡ内のリンクと、比較的低い得点０．８４の関連性を有する。得点が低い理由は、リンク７とフォルダＡの結びつきが間接的なものであり、いくつかの共通フォルダを

介するパスを必要とする（リンク３とフォルダＡ間の直接的な結びつきではない）ことである。そうはいっても幾分かの結びつきはあるので、得点は０より大きくなっている。ユーザがフォルダ内のリンクに関連するリンクを提示される際の提示の順序は得点に基づき、高得点ほど先に表示される。得点が十分高いリンクだけを表示するように任意の得点切り捨てがある。代替的に、リンクを得点の降順に表示して、より低い得点のリンクについて閲覧をいつストップするべきかはユーザが決するようにしてもよい。

【００３８】

このように、本発明は、特定のリンクが所与のフォルダ中のリンクにどの程度密接に関連するかを決定する。（リンクのコレクションを含む）特定のフォルダが別のフォルダにどの程度関連するかを決定するのに、このアプローチを変形して用いる。見本フォルダを指定して、システムはデータベース中のよく一致するフォルダを取得する。各フォルダ毎に、採点処理を適用し、見本フォルダに対する各リンクの得点を決定する。候補フォルダに対する総合得点をひとつにあわせて全体フォルダの得点とする。候補フォルダの平均得点を計算する。全フォルダにつき計算が終わると、最高総合平均得点が見本フォルダに最もよく一致するフォルダを示す。代替的に、高得点リンクを多く含むフォルダに高い値を、関連リンクの数が少ないフォルダに低い値をつける重みづけした平均得点を計算してもよい。

【００３９】

図７に示すのは、判定されたフォルダ間関係の例である。例示のテーブルはフォルダ間の共通・共有のリンクを介した最短パスを示す。図中フォルダＡはフォルダＦに３リンクパスで結びついている。すなわちフォルダＡとフォルダＦの間の結びつきは、間接的なものである。これらのフォルダには両方に共通するリンクは存在しない。ＡからＦにたどり着くには、共通のリンクを有する他の２つのフォルダを経由する必要がある。このことは、フォルダＡ及びＦがおそらくはそれほど高い関連性を持たないことを示している。

【００４０】

図８は、関連ブックマークをマッチングさせる、本発明のマッチング機能のフ

ローチャートを示す。ステップ８００では、適当な変数すべてを初期化する（例えば、マッチカウントや、ｉ）。所与の見本フォルダ内の全リンクに対するループ処理、処理対象ブックマークへのインデックス：変数ｉ（ステップ８０２）。ｉ番目ブックマークに関連するブックマークを関連テーブルから結果配列へ加える（ステップ８０４）。関連ブックマークのマッチカウントを増分する（ステップ８０６）。マッチした数の合計を記録する（ステップ８０８）。フォルダ内にブックマークが残っているかをチェックする（ステップ８１０）。残っていれば、処理はステップ８０４に戻り次のブックマークを処理する。残っていなければ、最終マッチセットをマッチカウントでソートし（ステップ８１２）、結果をディスプレイ表示する（ステップ８１４）。

【００４１】

図９に示すのは、本発明による関連リンク処理の例示的フローチャートである。初期化（９００）後、最初のブックマークフォルダを取得する（９０２）。ファイルの終わり（EOF）に達したら（９０４）、終了する（９１２）。当該フォルダ内のすべてのブックマークのペア（ｉ，ｊ）に対して以下のループ処理をおこなう（９０６）、そのペアに係る関連性カウントを増分する（９０８）。これを、当該特定のフォルダ内にあるすべてのブックマークに対して行う（９１０）。一つのフォルダ内の全ブックマークに対し処理したら、プログラムは次のフォルダの処理に移り（９０２）、全フォルダに対し処理を終了する。

【００４２】

図８のマッチング機能と図９の関連リンク処理を含む本発明の例示的なＪａｖａ実行形式を補遺Ｉとして添付する。

【００４３】

この処理の結果は、リンクのリストと、図６との関連で説明した得点である。得点が高ければ高いほど、リンクはフォルダデータベース内のリンクの関連クラスターに対してより密接に関連していた。補遺Ｉのアルゴリズムからの出力例を補遺IIとして添付する。

【００４４】

要するに、本発明のシステムはフォルダのリンクを関連づけるユーザのコンセ

ンサスによってマッチングがなされる上述の従来型自動ウェブサーチエンジンの欠点を克服するのに役立つ。書店のリンクを合衆国政府に関するリンクのコレクションと一緒に配置しているユーザがあったとしても、（もちろん、自身のリンクフォルダ中にそのような関連づけをする者が多ければ、その書店と例えばホワイトハウスとの間に論理的結びつきが多少あることを意味するので、そのような事情がなければ）重みづけ機能を利用しているのでそのような関連は拒絶される。

【００４５】

前記のフォルダに基づく自動カテゴリ分け及び個人ユーザのリンクディレクトリは、理解しやすいウェブディレクトリの作成に役立つ。上述のように、ユーザは自分の個人フォルダ階層中にリンクをファイルする。マッチング及びカテゴリ分け処理は、この情報を使って、関連するリンク及びフォルダを取得する。この基本メカニズムを拡張して、グローバルウェブリンクディレクトリを作成するメカニズムを形成する。処理は次のように進行する。実際にすべての個人ユーザフォルダは、マスタフォルダやリンクディレクトリの内容と重なっている部分をもっている。例えば、かなり完成度の高いマスターリンクディレクトリがすでに存在する場合、ユーザが自分の個人フォルダに新規リンクを追加するとき、フォルダマッチング処理を利用して、どのマスタフォルダがそれを保存する最良の場所かを決定する。これは、当該リンクを含む個人ユーザフォルダすべてを検索することによって、実行される。ユーザがそのリンクを自身の個人リンクコレクションに追加するとき、システムは、同一フォルダ中の他のリンクに基づいて、どの他のリンクが最も密接に関連しているかを「学習」する。そのリンクを含む個人ユーザフォルダが多いほど、関連するリンクを取得するのに利用できるより多くのデータが存在することになる。この情報によって、処理はマスターリンクディレクトリを検索し、そのリンクを含む個人ユーザフォルダと最もよく一致するフォルダを取得する。このリンクは、最も密接な関連性を有するリンクを含む一つ以上のマスタフォルダに追加される。

【００４６】

これは、マスターリンクディレクトリ中のフォルダにリンクを追加する処理で

ある。マスターリンクディレクトリのフォルダ階層の順序を提案するために、基本的なマッチング処理も利用される。すなわち、サブフォルダを配置するフォルダ群が決定する。結果はリンクを保存する領域のツリー構造となる。例えば、「園芸」という名前のフォルダが「野菜」及び「花」という名前のサブフォルダを含む場合、個人ユーザの私的組織化構造に基づいて、どうすればマスターリンクディレクトリをこのように組織化することができるのか？が問題となる。マスターリンクフォルダに最もよく一致するユーザリンクフォルダを決定するために、マッチングが使用される。全ユーザを横断してマッチングを実行してから、ユーザの個人リンクフォルダを調べる。多くのユーザを横断して、共通親フォルダに含まれる、例えば、花のリンクフォルダと一致するとともに、ほとんどの場合、野菜のリンクフォルダと一致するリンクコレクションが取得される。したがって、システムはユーザの個人リンクフォルダを精査し、その分類法を学習する。

【００４７】

リンクカテゴリ化システムのユーザにパブリックリンクの有用なコレクションを作成するインセンティブを提供するため、代替的実施態様では、中央サーバで報酬インセンティブ計画を実行することができる。上述したように、システムは、パブリックリンクフォルダを参照するユーザの数を追跡している。この参照数カウントメカニズムによって、アクセス統計（パブリックフォルダ内のリンクを人々がクリックする頻度）とともに、パブリックフォルダの人気度を位置づけるための計算式が開発された。単純なアプローチとしては、読み出し専用の参照の数に基づきフォルダを順序づける方法がある。他のユーザからの読むだけの参照が多ければ多いほど、高得点となる。最高得点をあげたフォルダ制作者は全体リンクコレクションへの貢献に対し報酬を受け取ることができる。この報酬の方式は、自分のコレクションに対して受ける報酬額を増やすために自分自身の個人リンクコレクションについて評判を拡めようとするので、このサービスの普及促進にも役立つ。報酬決定のための単純な参照数カウント方式に対して修正を施し、ある期間内の参照総数及びコレクションをクリックした数の両方をまぜて計算する関数をもたせることもできる。利用可能な総報酬資金に基づき、等級要因を選択し、２つのパラメータの一次結合を利用する。

【００４８】

ウェブ上の「コミュニティ」の展開は、非常に興味をそそる分野である。ウェブコミュニティとは、共通の関心事をわかちあう同じウェブサイトに集う人々のグループである。これを達成するために、特定の話題に関するディスカッショングループを立ち上げる等、様々なアプローチがとられている。例えば、Ｅトレード（Etrade）のようなサイトには、個人の株式と投資のアイデアに関するディスカッショングループがある。この方法には、コミュニティ構築者によって選択されたトピック群に過度に特化してしまう等、いくつかの欠陥がある。ユーザが、より抽象的な関心事に関するコミュニティを効果的に構築することができる方法はなかった。これを克服するために、カテゴリ化とマッチングの機能を使用することができる。この機能は、類似リンクコレクションを有する人々を発見するために利用できる。類似リンクコレクションを有しているということは、その類似リンクコレクションのコンテンツが反映する関心事をその人たちが共有していることを示す強力な指標である。例えば、ユーザは「園芸」に関するリンクのフォルダを作成する。システムは、そのようなリンクコレクションを有する他のユーザを見つけて、彼らに、似たような園芸フォルダを有するすべてのユーザの、動的に作成されたディスカッショングループに参加するよう呼びかける。ユーザの全体コミュニティへの新参者が、その園芸フォルダと一致するフォルダを作成すると、彼らにも、グループ参加を呼びかける。上述のコミュニティ参加を人々に呼びかけるための、このような処理は、プライバシーの不安を克服するものである。ユーザは、コミュニティに参加するよう誘われる。参加するまでは、他のユーザに彼らの存在が知られることはない。同様に、ディスカッショングループを、任意のパブリックフォルダ内で提供する選択肢もある。パブリックフォルダの所有者がそうすることに決めたら、彼又は彼女はディスカッション機能を有効にすることができる。これによって、パブリックフォルダにアクセスする者なら誰でも、パブリックフォルダリンクとともに表示されることになるメッセージを投稿することができるようになる。パブリックフォルダにアクセスする人々のグループは、パブリックフォルダ内のリンクによって参照されるコンテンツが反映する関心事を共有している可能性が高いので、その特定の関心事に関して展開する

コミュニティを作成することは容易となる。

【００４９】

多様な実施態様の代替的使用には、ウェブ広告のターゲットを絞るためにリンクを関連づけ、採点することも考えられる。これを達成するために、潜在的広告主は、市場開拓すべき製品又はサービスを代表するものと考えられるリンクコレクションを作成する。例えば、ある広告主が園芸用品を販売しているとすると、園芸のリンクのコレクションが収集され、見本リンクフォルダに配置される。フォルダのマッチング及びカテゴリ化の処理を用いて、一致するユーザリンクフォルダが取得される。すると、広告主のウェブサイトへのリンクが、自動的に、そのユーザのリンクコレクションに追加され、そのユーザがそのフォルダに入るたびに現れるようになる。同一フォルダに複数の広告主のリンクが挿入されることも可能であり、すべての園芸広告主がシステム中の全園芸関連ウェブフォルダ中にそのウェブサイトのＵＲＬへの参照を挿入することができる。ユーザが広告主からの見本と一致するフォルダに入るたびにページのトップにバナー広告を表示するために、同様のアプローチが利用される。

【００５０】

ＣＧＩスクリプト（動的なウェブページを生成するために使用される公知の方法）を使用して、ユーザは、任意のウェブページ上に、動的ホットリストを挿入することによって、埋め込みリンクリストを作成することができる。埋め込みリンクリスト機能の Java によるインプリメンテーションの例を補遺IIIとして添付する。リモートコンピュータで走るアクセスモジュールによって、中央サーバへ接続する。このサーバは、リンクフォルダのコンテンツをリモートのアプリケーションに返す。すると、そのフォルダは、ホットリストとして、ウェブページに表示される。このアプローチはホットリストを動的にする。フォルダに追加や変更があれば、そのウェブページに次にアクセスした際、新規の追加又は変更が表示される。マッチングとカテゴリ化の機能によって、そのホットリストには、他の類似リンクが補充され、特定のフォルダ内のリンクのみならず、密接に関連するリンクも含まれる。

【００５１】

本発明について、その具体的な実施態様と適用例に関して、例示及び説明を加えてきた。本発明の検討を容易ならしめるために好ましい実施態様を一応仮定してはいるが、上述の実施態様は単に本発明の原理の例示にすぎないのであって、本発明の実施態様をそれに限定する趣旨ではない。本発明の様々な便益を実現するために、ここに一つ一つ列挙して開示した実施態様及び教示内容に変形を施した代替的実施態様が推論及び実行可能であることは、当業者の当然理解するところである。

【００５２】

さらに、当業者によれば、本発明の趣旨及び範囲を逸脱せずに、前述の、及び、その他多様な変更、省略、追加を考案できるものとする。したがって、本発明は開示された実施態様に限定されず、添付の請求の範囲によって画定されることが予定されている。

【００５３】

補遺　Ｉ：サンプル Java ソースコード

APPENDIX I: Sample Java Source Code

```
// Java class function to analyze a series of link
// directors
// Author: David Siegel
//

import java.util.*;
import java.io.*;
import java.lang.*;

import Folder;
import Directory;
import Permutation;

class Analyze {
  Directory _dir;
  Folder _f;
  double[] _results;

  // Constructor that creates an instance of the analyze
  // class.  We pass in the directory tree that contains
  // all link folders in the system and a folder to be
  // matched against the directory.
  public Analyze(Directory dir, Folder f) {
    _dir = dir;
    _f = f;
    _results = new double[1000];
    int i;
    for (i=0; i<1000; i++)
      _results[i] = 0.0;
  };

  // Compute all combinations of the folder given in the
  // analyze contructor and run the matching algorithm on
  // each.  We take all combinations of m links taken n
  // at a time, where m is the number of links in the
  // exemplar folder and n ranges from m to 1.
  public Folder permute(double thresh, int level) {
    int[] x = _f.toArray();
    int n = _f.count();
    int i;

    // This bootstraps the combination process by calling
    // the combination helper routine ones for each m
    // (where m is the number of items being considered
    // in the combination, as described above.
    for (i=n-1; i>0; i--) {
      int[] p = new int[i];
      doPermute(p, x, n, i, i);
    }

    Folder folder = new Folder();

    // We build a folder here with the summary results
    // from all matches that we found by running the
    // above permutation.  Note that we only include
    // matches that exceed a given score threshold, as
    // specified in an argument to this routine.  We
    // weight the score by the "level", where level is
    // the number of times the routine was reinvoked on
```

【００５４】

補遺　Ｉ：サンプル Java ソースコード

APPENDIX I: Sample Java Source Code

```
    // its own results.  This is how indirect
    // relationships are found.
    for (i=0; i<1000; i++) {
      double v = _results[i] / (double)level;
      if ((v > 0) && !_f.contains(i)) {
        System.out.println(i + " ---> " + v);
        if (v > thresh)
          folder.addLink(i);
      }
    }

    return folder;
  };

  // This method does the work of the matching process.
  // The actual combinations m at a time are computed
  // here, recursively.  The cumulative result score is
  // also computed.
  //
  // Arguments to the routine:
  // p: accumulated permutations
  // x: input array
  // n: length of input array
  // m: remaining count to permute
  // c: total length of permutation
  private void doPermute(int[] p, int[] x, int n,
              int m, int c)
  {
    int i, j;

    // Create a copy of the input link list omitting one
    // link at a time.  This is done to compute the
    // combinations of links.
    for (i=0; i<n; i++) {
      int[] xCopy = new int[n-1];
      for (j=0; j<n-1; j++) {
        if (j<i)
          xCopy[j] = x[j];
        else
          xCopy[j] = x[j+1];
      }

      // Build the new combination of links to be
      // considered.
      p[m-1] = x[i];

      // If the length of the remaining links to be added
      // to the combination is greater than one, invoke
      // the routine recursively to add the remaining
      // links to the combination.  If not, we've got a
      // combination and are ready to compute a score.
      if (m > 1) {
        doPermute(p, xCopy, n-1, m-1, c);
      } else {
        // Build a link folder from the combination.
        Folder f = new Folder(p, c);

        // Match this folder against the directory tree,
        // looking for all folders that contain this
```

補遺　Ｉ：サンプル Java ソースコード

APPENDIX I: Sample Java Source Code

```
        // exact set of links.
        Directory.Result rc = _dir.findSimilar(f);

       // Create a score weighting factor. Here, c is the
      // total length of the permutation and _f.count() is
      // the total length of the initial exemplar
      // provided.  Thus, the more links from the original
      // exemplar provided that are in this combination,
      // the greater the score.
         double factor = (double)c/(double)_f.count();

         // Now scan all matches and record the scores for
         // each link. Only keep the highest score, since
         // the link could be find in more than one of the
         // combination passes through this routine.
         int k;
         for (k=0; k<1000; k++) {
           double v = (rc.folders[k]*factor)/rc.total;
           if (v > _results[k])
             _results[k] = v;
         }
       }
     }
   };

   // Print a summary of the results.
   public int printStats(Folder folder, int results[],
      int total, double factor)
   {
     int i;
     for (i=0; i<1000; i++) {
       if (results[i] > 0)
         System.out.println("Found '" + i + "' " +
           results[i] + " times (" +
           ((float)results[i]*factor)/(float)total + ")");
     }
     return total;
   };

   // Helper function to print an array.
   public void printArray(int[] array, int count) {
     int i;
     for (i=0; i<count; i++)
       System.out.print(array[i] + " ");
     System.out.println();
   };
}
//
// Java functions to manipulate a system wide directory
// of link folders.
// Author: David Siegel
//

import java.util.*;
import java.io.*;
import java.lang.*;

import Folder;
```

補遺　Ｉ：サンプル Java ソースコード

APPENDIX I: Sample Java Source Code

```
class Directory {
  private Folder[] _folders;
  private int _allocated;
  private int _inUse;

  // Create an empty directory (limitted to 1000 link folders for now).
  public Directory() {
    _folders = new Folder[1000];
    _allocated = 1000;
    _inUse = 0;
  };

  // Add a folder from a string to the system wide folder
  // directory.
  public int addFolder(String s) {
    Folder f = new Folder();
    f.addLinks(s);
    return addFolder(f);
  };

  // Add a folder from a folder class to the system wide
  // folder directory.
  public int addFolder(Folder f) {
    _folders[_inUse] = f;
    return _inUse++;
  };

  // Count the number of times all links in an exemplar
  // folder match against folders in the system wide
  // directory.
  public int match(Folder exemplar) {
    int count = 0;
    int i;

    for (i=0; i<_inUse; i++) {
      if (_folders[i].match(exemplar))
        count++;
    }

    return count;
  };

  // Helper class that is the return value of the next
  // function.
  class Result {
    int[] folders;
    int total;
  };

  // This function takes an exemplar folder as an input,
  // and return an array that counts how many times each
  // link in the exemplar appears in the directory.  For
  // example, folders[i] of the result contains the count
  // of how many times link i matched against all the
  // links in the directory.
  public Result findSimilar(Folder exemplar) {
    int results[] = new int[1000];
    int matches = 0;
    int i;
```


【００５７】

補遺　Ｉ：サンプル Java ソースコード

APPENDIX I: Sample Java Source Code

```
    // Initalize the return result counts to 0.
    for (i=0; i<1000; i++)
      results[i] = 0;

    // For each folder in the system, see if we have
    // matches against the exemplar.  If we do, add one
    // to the count for a match for a link each time a
    // match has been found.
    for (i=0; i<_inUse; i++) {
      if (_folders[i].match(exemplar)) {
        matches++;
        Enumeration elems = _folders[i].keys();
        while (elems.hasMoreElements()) {
          Integer link = (Integer)elems.nextElement();
          if (!exemplar.contains(link))
            results[link.intValue()]++;
        }
      }
    }

    // Build and return the results.
    Result rc = new Result();
    rc.folders = results;
    rc.total = matches;
    return rc;
  };
}
```

【００５８】

補遺　Ｉ：サンプル Java ソースコード

APPENDIX I: Sample Java Source Code

```
//
// Java functions to manipulate a link folder.
// Author: David Siegel
//

import java.util.*;
import java.io.*;
import java.lang.*;

class Folder {
  private Hashtable _folder;

  // Constructor to create an empty link folder.
  public Folder() {
    _folder = new Hashtable();
  };

  // Constructor to create a link folder from a string
  // listing the links in the folder.
  public Folder(String linkString) {
    _folder = new Hashtable();
    addLinks(linkString);
  };

  // Constructor to create a link folder from an array,
  // where only the links that have a score greater than
  // the given threshold are added to the link folder.
  public Folder(int[] linkArray, int divisor,
    double thresh)
  {
    int i;
    _folder = new Hashtable();
    for (i=0; i<1000; i++) {
      if ((double)linkArray[i]/(double)divisor > thresh)
        _folder.put(new Integer((int)i), "");
    }
  };

  // Constructor to create a link folder from an array.
  public Folder(int[] array, int count) {
    int i;
    _folder = new Hashtable();
    for (i=0; i<count; i++) {
      _folder.put(new Integer((int)array[i]), "");
    }
  };

  // Return the links in the folder.
  public Enumeration keys() {
    return _folder.keys();
  };

  // Add a bunch of links in string format to a folder.
  public int addLinks(String linkString) {
    Reader in;
    in = new StringReader(linkString);
    StreamTokenizer parser = new StreamTokenizer(in);

    try {
```

補遺　Ｉ：サンプル Java ソースコード

APPENDIX I: Sample Java Source Code

```
    while (parser.nextToken() !=
                              streamTokenizer.TT_EOF)
    {
      switch (parser.ttype) {
      case StreamTokenizer.TT_EOL:
        break;
      case StreamTokenizer.TT_NUMBER:
        _folder.put(new Integer((int)parser.nval), "");
        break;
      }
    }
  } catch (IOException e) {
    return -1;
  }

  return 0;
};

// Method to add a link to the folder.
public void addLink(int link) {
  _folder.put(new Integer((int)link), "");
}

// Determine if the folder contains a given link.
public boolean contains(int link) {
  return _folder.containsKey(new Integer(link));
};

// Same as above for Integer type.
public boolean contains(Integer link) {
  return _folder.containsKey(link);
};

// Determine if an example folder matches the given
// folder.
public boolean match(Folder exemplar) {
  Enumeration elems = exemplar._folder.keys();
  boolean found = true;

  // Iterate through all the links in the exemplar and
  // make sure each and every one is in the folder.
  while (elems.hasMoreElements()) {
    Integer link = (Integer)elems.nextElement();
    if (!contains(link.intValue())) {
      found = false;
      break;
    }
  }

  // Return true if we have a match.
  return found;
};

// Determine if an example folder matches the given
// folder.
public void print() {
  Enumeration elems = _folder.keys();
  while (elems.hasMoreElements()) {
    Integer link = (Integer)elems.nextElement();
```

補遺　Ｉ：サンプル Java ソースコード

APPENDIX I: Sample Java Source Code

```
      System.out.print(link + " ");
    }
    System.out.println();
  };

  // Convert to an array.
  public int[] toArray() {
    int[] array = new int[_folder.size()];
    Enumeration elems = _folder.keys();
    int i = 0;
    while (elems.hasMoreElements()) {
      Integer link = (Integer)elems.nextElement();
      array[i++] = link.intValue();
    }
    return array;
  };

  // Return the number of links in the folder.
  public int count() {
    return _folder.size();
  };
}
```


【００６１】

補遺　Ｉ：サンプル Java ソースコード

APPENDIX I: Sample Java Source Code

```
//
// Main test harness for the link matching algorithms.
// Author: David Siegel
//

import java.util.*;
import java.io.*;
import java.lang.*;

import Folder;
import Directory;
import Analyze;

class Main {
  public static void main(String argv[]) {
    Directory dir = new Directory();

    // Add a bunch of folders to the system directory.
    // Each number represents a link number.  Each call
    // to addFolder adds a new folder.  For the purposes
    // of this test we do not track the folder directory
    // structure or the folder owner.
    dir.addFolder("1 2 3 5");
    dir.addFolder("1 2 3 5");
    dir.addFolder("1 2 3 5");
    dir.addFolder("1 2 3 7");
    dir.addFolder("2 2 3 7");
    dir.addFolder("2 2 3 9");
    dir.addFolder("2 2 3 5");
    dir.addFolder("1 5 9 10");
    dir.addFolder("1 5 9 10");
    dir.addFolder("1 5 9 11");
    dir.addFolder("1 5 9 11");
    dir.addFolder("1 5 9 12");
    dir.addFolder("1 5 9 12");
    dir.addFolder("1 5 9 12");
    dir.addFolder("1 5 9 12");
    dir.addFolder("1 5 9 12");
    dir.addFolder("1 5 9 12");
    dir.addFolder("10 11 12 44");
    dir.addFolder("12 15 19 23");
    dir.addFolder("12 15 19 22");
    dir.addFolder("12 15 19 22");
    dir.addFolder("12 15 19 22");
    dir.addFolder("12 15 19 22");
    dir.addFolder("12 15 19 22");
    dir.addFolder("12 15 19 22");
    dir.addFolder("12 15 19 22");
    dir.addFolder("12 15 19 22");

    // Perform the matching process with the specified
    // exemplar folder.  Thus, the input to the matching
    // process is the system wide folder directory that
    // was created above, plus an exemplar folder.  The
    // goal is to find links that are related to the
    // input exemplary.
    System.out.println("Level 1:");
    Analyze study = new Analyze(dir,
                          new Folder("1 2 3 4 5 9"));
```


【００６２】

補遺　Ｉ：サンプル Java ソースコード

APPENDIX I: Sample Java Source Code

```
    Folder f = study.permute(0.01, 1);

    // We now reinvoke the matching algorithm on the
    // results of the first match.  This is in essence a
    // level 2 match which finds indirectly related links
    // to the initially given exemplar.  This process
    // could be repeated to find level 3, 4, etc.
    // relationships, though after a while the
    // relationships become too distant to be meaningful.
    System.out.println("Level 2:");
    study = new Analyze(dir, f);
    study.permute(0.01, 2);
  };
}
```

補遺　Ｉ：サンプル Java ソースコード

APPENDIX I: Sample Java Source Code

```
/*    $Id: RelatedUrl.java,v 1.4 Exp $    */
// Author: David Siegel
//
package BlinkUtil;

import java.io.*;
import java.lang.*;
import java.util.*;
import java.util.Date;
import java.text.*;
import java.sql.*;
import java.sql.SQLException;

public class RelatedUrl extends Url {

  private int _matchCount;

  public void finalize() throws Throwable {
    super.finalize();
  }

  public RelatedUrl(Session session, ResultSet rs) throws SQLException {
    super(session, rs);
    _matchCount = rs.getInt("match_count");
  }

  public int matchCount() {
    return _matchCount;
  }

  public void addMatchCount(RelatedUrl rurl) {
    _matchCount += rurl._matchCount;
  }
}
```

補遺　Ｉ：サンプル Java ソースコード

APPENDIX I: Sample Java Source Code

```
/*    $Id: RelatedUrlDir.java,v 1.7 Exp $ */
// Author: David Siegel
//
package BlinkUtil;

import java.io.*;
import java.lang.*;
import java.util.*;
import java.util.Date;
import java.text.*;
import java.sql.*;
import java.sql.SQLException;

public class RelatedUrlDir {

  Session _session;
  Hashtable _ht;
  RelatedUrl[] _rurls;

  public void finalize() throws Throwable {
    super.finalize();
    _session = null;
    _ht = null;
    _rurls = null;
  }

  public RelatedUrlDir(Session session) {
    _session = session;
    _ht = null;
    _rurls = null;
  }

  public Session session() {
    return _session;
  }

  public RelatedUrl[] items() {
    return _rurls;
  }

  public int size() {
    if (_rurls == null)
      return 0;
    else
      return _rurls.length;
  }

  public void add(ResultSet rs) throws SQLException {
    RelatedUrl next = new RelatedUrl(_session, rs);
    RelatedUrl rurl = (RelatedUrl)_ht.get(new Integer(next.urlId()));
    if (rurl == null)
      _ht.put(new Integer(next.urlId()), next);
    else
      rurl.addMatchCount(next);
  }

  public void update(int dirId) throws SQLException, FatalException {
    _ht = new Hashtable();
```

補遺　Ｉ：サンプル Java ソースコード

APPENDIX I: Sample Java Source Code

```
    SQL.RelatedUrlDir(this, dirId);

    if (_ht.size() > 0) {
      _rurls = new RelatedUrl[_ht.size()];
      Enumeration elems = _ht.elements();
      int i = 0;
      while (elems.hasMoreElements()) {
      RelatedUrl rurl = (RelatedUrl)elems.nextElement();
      _rurls[i++] = rurl;
      }
      sort();
    } else
      _rurls = null;
  }

  private void sort() {
    int i, j;
    for (i = 0; i < _rurls.length-1; i++) {
      for (j = i+1; j < _rurls.length; j++) {
      if (_rurls[i].matchCount() < _rurls[j].matchCount()) {
        RelatedUrl swap = _rurls[i];
        _rurls[i] = _rurls[j];
        _rurls[j] = swap;
      }
      }
    }
  }
}
```


【００６６】

補遺　Ｉ：サンプル Java ソースコード

APPENDIX I: Sample Java Source Code

```
/*$Id: RelatedUrlGenerate.java,v 1.5 Exp $      */
// Author: David Siegel
//
package BlinkUtil;

import java.io.*;
import java.lang.*;
import java.util.*;
import java.util.Date;
import java.text.*;
import java.sql.*;
import java.sql.SQLException;

public class RelatedUrlGenerate {

  class Item {
    private int _urlId;
    private int _count;

    public Item(int urlId) {
      _urlId = urlId;
      _count = 0;
    }

    public int urlId() { return _urlId; }
    public int count() { return _count; }
    public void increment() { _count++; }
  }

  Hashtable _ht;

  public RelatedUrlGenerate() {
    _ht = null;
  }

  public void add(ResultSet rs) throws SQLException {
    int matchUrlId = rs.getInt("url_id");
    Item item = (Item)_ht.get(new Integer(matchUrlId));
    if (item == null) {
      item = new Item(matchUrlId);
      _ht.put(new Integer(matchUrlId), item);
    }
    item.increment();
  }
```

補遺　Ｉ：サンプル Java ソースコード

APPENDIX I: Sample Java Source Code

```
  public void update(int urlId) throws SQLException, FatalException {
    _ht = new Hashtable();

    SQL.RelatedUrlFind(this, urlId);

    Enumeration elems = _ht.elements();
    while (elems.hasMoreElements()) {
      Item item = (Item)elems.nextElement();
      SQL.RelatedUrlUpdate(urlId, item.urlId(), item.count());
    }
  }
}
```


【００６８】

補遺　II：出力例

APPENDIX II: Sample Output

```
Level 1:
7 ---> 0.125
10 ---> 0.1
11 ---> 0.1
12 ---> 0.3
Level 2:
1 ---> 0.08333333333333333
2 ---> 0.125
3 ---> 0.125
5 ---> 0.08333333333333333
9 ---> 0.08333333333333333
15 ---> 0.0703125
19 ---> 0.0703125
22 ---> 0.0625
23 ---> 0.0078125
44 ---> 0.375
```

【００６９】

補遺　III：サンプル Java ソースコード

APPENDIX III: Sample Java Source Code

```
/*    $Id: Embed.java,v 1.7 Exp $    */
// Author: David Siegel
//
import com.blink.blink.Config;
import java.io.*;
import java.lang.*;
import java.util.*;
import java.net.MalformedURLException;
import java.sql.SQLException;
import javax.servlet.*;
import javax.servlet.http.*;

import BlinkUtil.*;

public class Embed extends HttpServlet {

  private static Session _session;
  private State _state;
  private int _count = 0;

  public void init(ServletConfig servletConfig) throws ServletException {
    super.init(servletConfig);
    try {
      Config.load();
    } catch (IOException e) {
      throw new ServletException("Can't load configuration data:" +
          e.getMessage());
    }
    Log.init();
    _state = State.getState();
    _session = new Session(null, null, null, null);
    Config config = Config.getConfig();
    _session.dispatch(config.get("html.url.server.root") +
      config.get("html.url.run"));
  }

  private String fmt(String html) {
    html = Page.quoteQuote(html);
    return "document.write(\"" + html + "\");\n";
  }

  public void service(HttpServletRequest req, HttpServletResponse res)
    throws IOException, ServletException {
    ServletOutputStream out = res.getOutputStream();
    res.setContentType("text/html");

    String cmd = req.getParameter("cmd");

    if (cmd.equals("gen")) {
      String dirIdStr = req.getParameter("dir");
      String options = req.getParameter("opt");
      try {
      EmbedLinks e = new EmbedLinks(_session);
      String str = e.html(dirIdStr, options);
      if (Log.Debug) Log.log(Log.Dispatch, fmt(str));
      out.println(fmt(str));
```

補遺　III：サンプル Java ソースコード

APPENDIX III: Sample Java Source Code

```
        } catch (Exception e) {
        out.println(fmt("Folder not available"));
        if (Log.Syslog) Log.log(Log.Embed, "Embed Folder(" + dirIdStr +
          ", " + options + ") got:" + e.getMessage());
        }
      } else if (cmd.equals("hit")) {
        String urlIdStr = req.getParameter("url");
        String dirIdStr = req.getParameter("dir");
        try {
        int urlId = Page.getId(urlIdStr, "url");
        int dirId = Page.getId(dirIdStr, "dir");
        Url url = _session.getUrl(urlId, dirId);
        url.hit(dirId);
        res.sendRedirect(url.href());
        } catch (Exception e) {
        out.println(fmt("Cannot launch URL"));
        if (Log.Syslog) Log.log(Log.Embed, "Embed hit(" + urlIdStr + "," +
          dirIdStr + ") got:" + e.getMessage());
        }
      }
    }
  }
```

【図面の簡単な説明】

【図１】　本発明の好適な実施態様としてのシステムアーキテクチャのインプリメンテーションを表す。

【図２】　本発明の好適な実施態様による主な処理ループのフローチャートを表す。

【図３】　本発明の好適な実施態様による例示的メイン画面を表す。

【図４】　図４Ａ及び４Ｂは、本発明の好適な実施態様におけるユーザ定義及びフォルダマッピングデータベースの図式を表す。

【図５】　本発明の好適な実施態様において例示的リンク数に対して導かれたフォルダマトリックスを表す。

【図６】　本発明の好適な実施態様におけるリンク採点を示すリンク採点テーブルを表す。

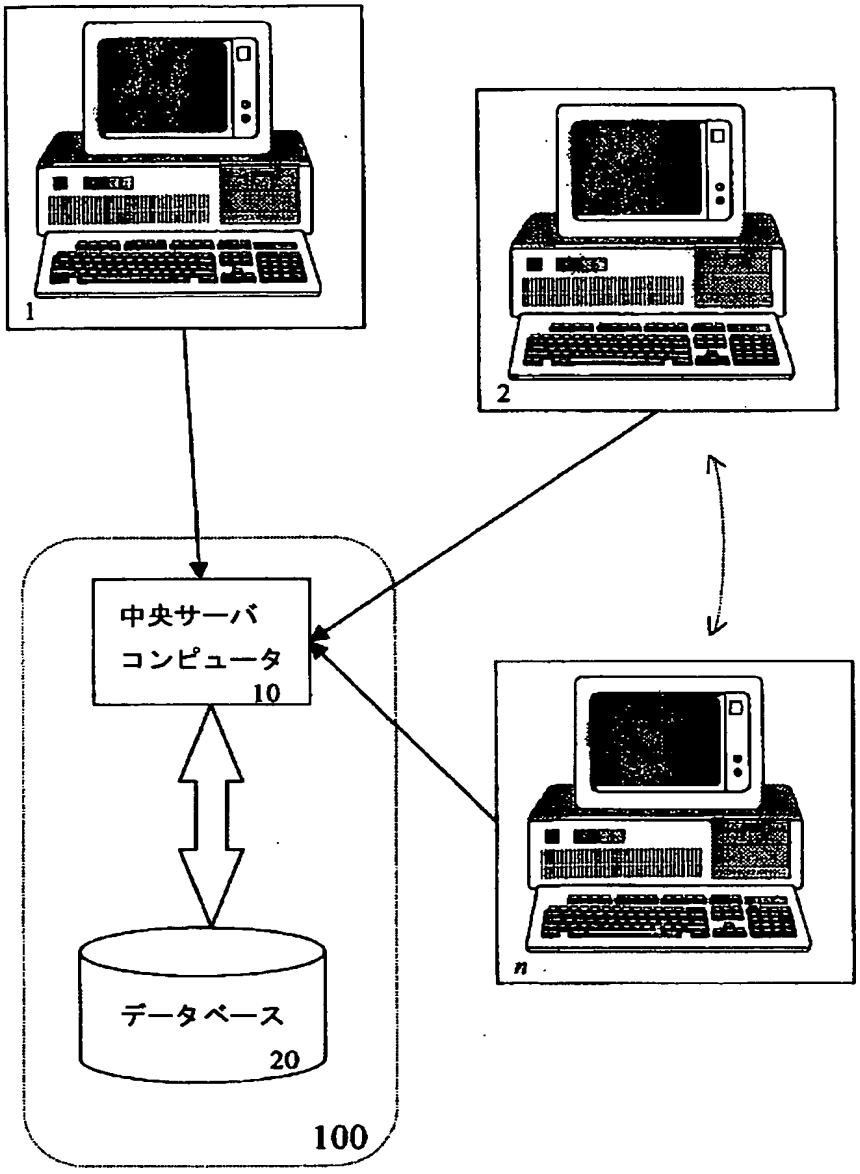
【図７】　本発明の好適な実施態様において導かれたリンク関係結果テーブルを

表す。

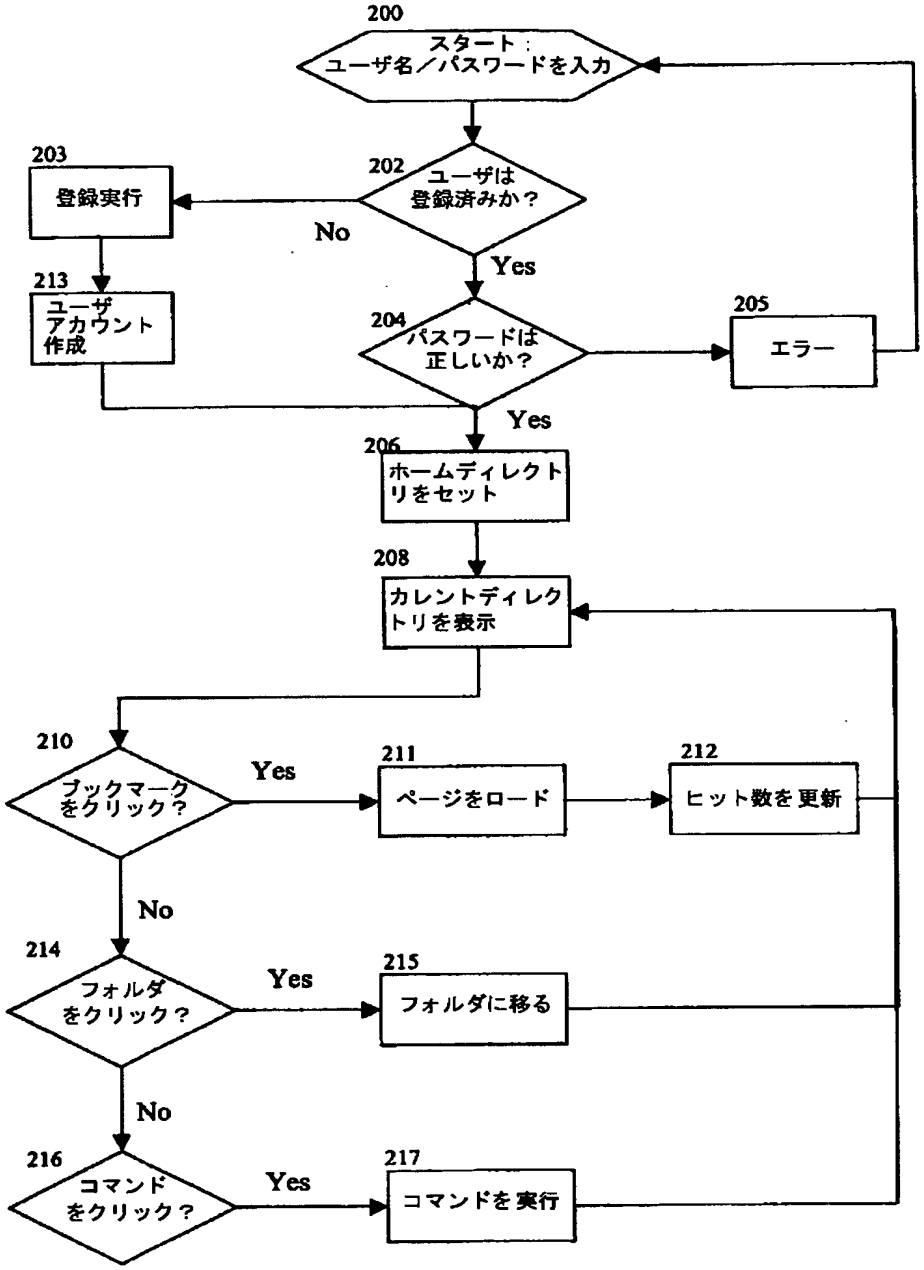
【図８】　本発明のマッチング機能の好適な実施形態のフローチャートを表す。

【図９】　本発明の関連リンク処理の好適な実施形態のフローチャートを表す。

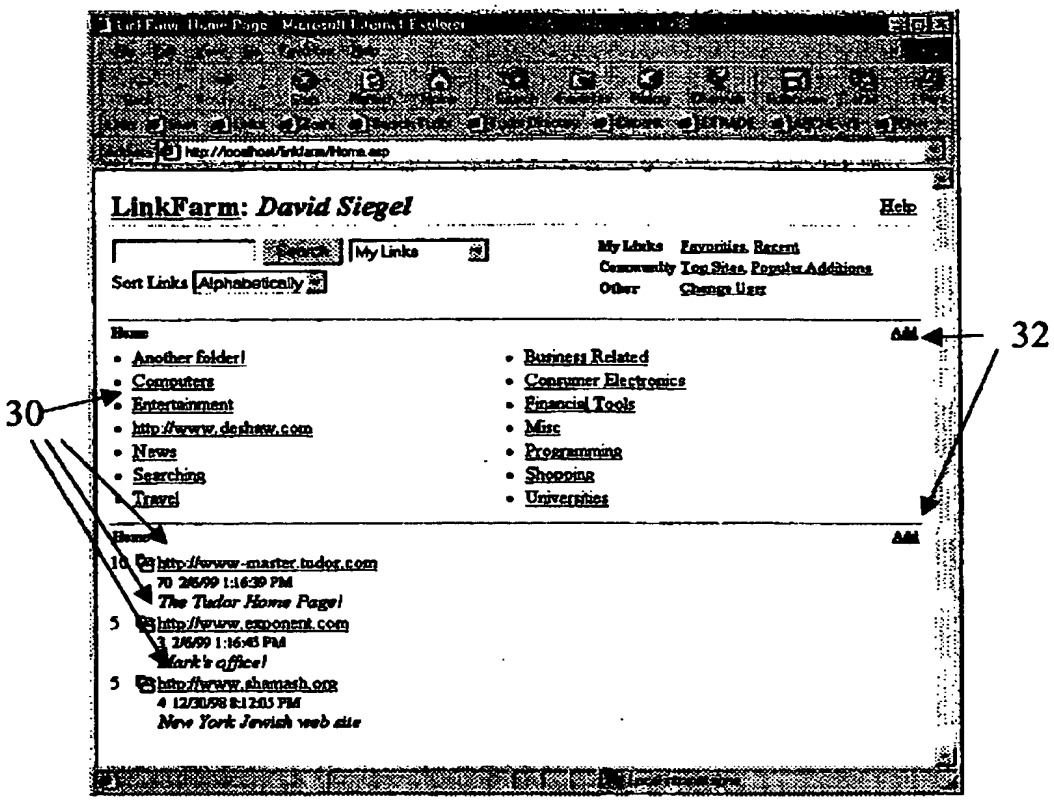
【図１】



【図２】


200
スタート：
ユーザ名／パスワードを入力
202
ユーザは
登録済みか？
No
203
登録実行
Yes
213
ユーザ
アカウント
作成
204
パスワードは
正しいか？
205
エラー
Yes
206
ホームディレクト
リをセット
208
カレントディレク
トリを表示
210
ブックマーク
をクリック？
Yes
211
ページをロード
212
ヒット数を更新
No
214
フォルダ
をクリック？
Yes
215
フォルダに移る
No
216
コマンド
をクリック？
Yes
217
コマンドを実行

【図３】


LinkFarm: David Siegel
Help
Search
My Links
Sort Links
Alphabetically
My Links
Favorites, Recent
Community
Top Sites, Popular Additions
Other
Change User
Home
Add
Another folder!
Computers
Entertainment
http://www.deshaw.com
News
Searching
Travel
Business Related
Consumer Electronics
Financial Tools
Misc
Programming
Shopping
Universities
Home
Add
http://www-master.tudor.com
70 2/6/99 1:16:39 PM
The Tudor Home Page!
http://www.exponent.com
3 2/6/99 1:16:45 PM
Mark's office!
http://www.shamash.org
4 12/30/98 8:12:05 PM
New York Jewish web site
30
32

【図４】

フィールド	説　明
ユーザＩＤ	固有ユーザ識別番号
ユーザ名	固有ユーザ名
名	名
姓	姓
職業	職業コード
年齢	年齢（又は年齢の範囲）
収入	収入

図4A

フィールド	説　明
フォルダＩＤ	固有フォルダ識別番号
ユーザＩＤ	フォルダ所有者のユーザＩＤ

図4B

【図５】

	A	B	C	D	E	F
1	X	X				
2	X	X	X			
3		X	X	X		
4				X	X	
5				X	X	X
6					X	X
7						X

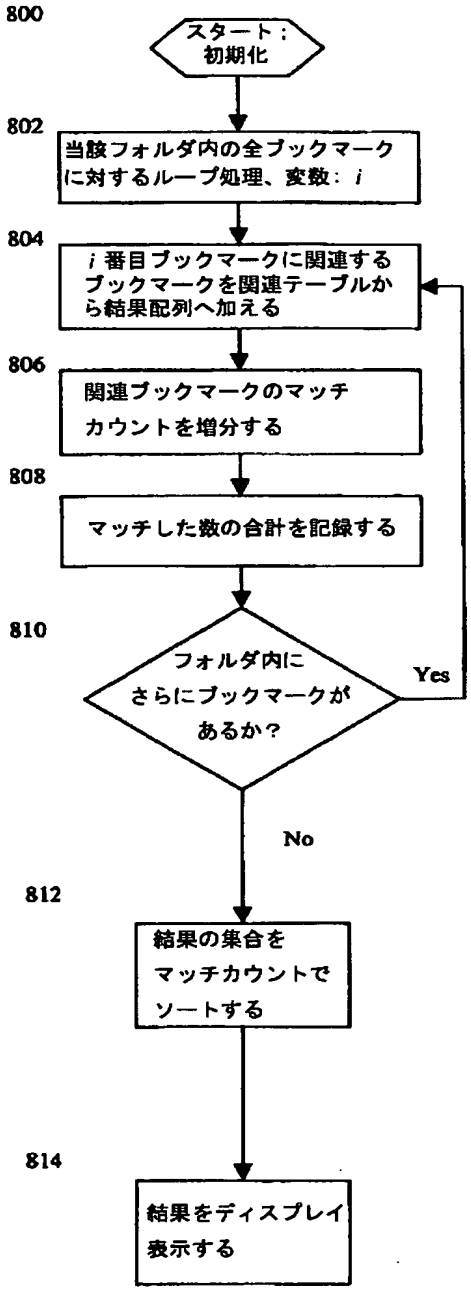
【図６】

	B	C	D	E	F	合計
2						
3	1	1	0.5	0.5	0	3
4	0	1	0.5	0.5	0	2
5	0	0	0.5	0.5	0	1
6	0	0	0.5	0.5	0.34	1.34
7	0	0	0	0.5	0.34	0.84

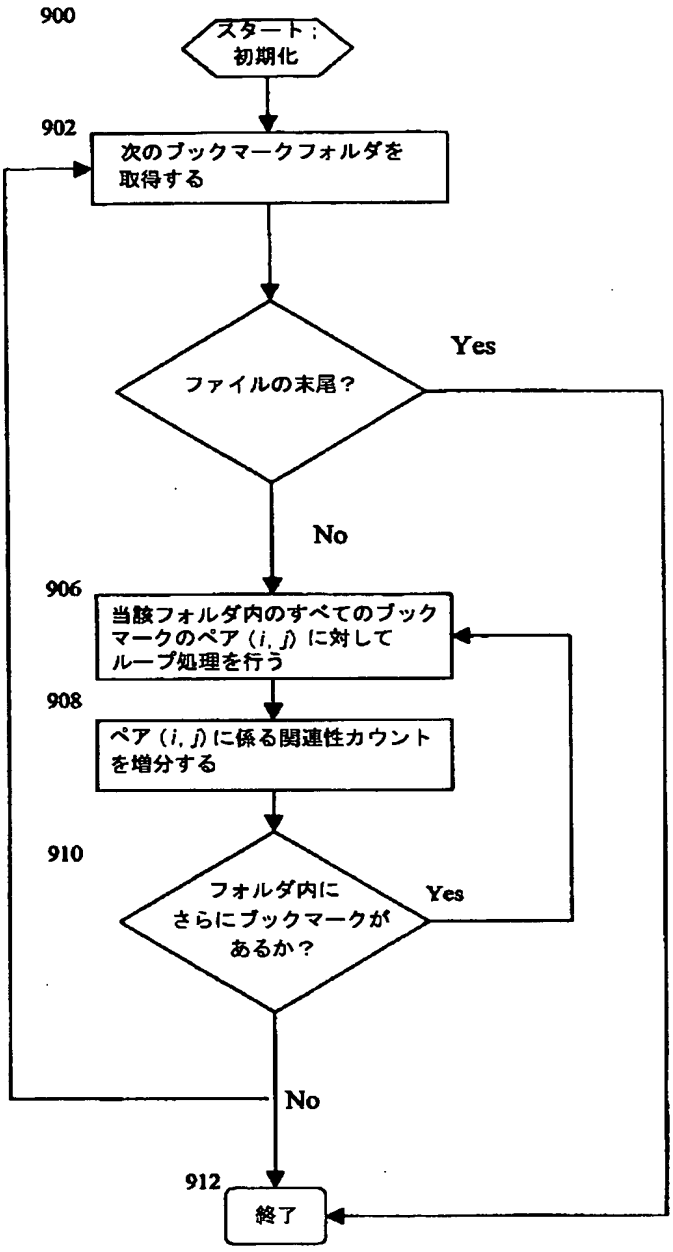
【図７】

	A	B	C	D	E	F
A	0					
B	1	0				
C	1	1	0			
D	2	1	1	0		
E	2	1	1	1	0	
F	3	2	2	1	1	0

【図８】


800
スタート：
初期化
802
当該フォルダ内の全ブックマーク
に対するループ処理、変数：i
804
i番目ブックマークに関連する
ブックマークを関連テーブルか
ら結果配列へ加える
806
関連ブックマークのマッチ
カウントを増分する
808
マッチした数の合計を記録する
810
フォルダ内に
さらにブックマークが
あるか？
Yes
No
812
結果の集合を
マッチカウントで
ソートする
814
結果をディスプレイ
表示する

【図９】


900
スタート；
初期化
902
次のブックマークフォルダを
取得する
ファイルの末尾？
Yes
No
906
当該フォルダ内のすべてのブック
マークのペア（i, j）に対して
ループ処理を行う
908
ペア（i, j）に係る関連性カウント
を増分する
910
フォルダ内に
さらにブックマークが
あるか？
Yes
No
912
終了

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/04588

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) :G06F 13/00, 13/14, 17/00, 17/30, 17/60
US CL :Please See Extra Sheet.
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 235/375; 345/329; 705/26, 27; 707/2, 3, 10, 100, 104, 200, 203, 506; 709/1, 201, 217, 218, 219, 235

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A, P	US 5,956,027 A (KRISHNAMURTHY) 21 SEPTEMBER 1999, col. 1-4.	1-41
A, E	US 6,052,687 A (MIURA ET AL.) 18 APRIL 2000, col. 4-12	1-41
A, E	US 6,064,979 A (PERKOWSKI) 16 MAY 2000, col.4-21,	1-41
A, E	US 6,052,717 A (REYNOLDS ET AL.) 18 APRIL 2000, col. 3-15.	1-41
A, P	US 6,023,701 A (MALIK ET AL.) 08 FEBRUARY 2000, col. 2-8	1-41
A, P	US 5,895,461 A (DE LA HUERGA ET AL.) 20 APRIL 1999, col. 4-12.	1-41

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 26 MAY 2000	Date of mailing of the international search report **06** JUL 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer GLEN BURGESS Telephone No. (703) 305-4792

Form PCT/ISA/210 (second sheet) (July 1998) ★

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/04588

A. CLASSIFICATION OF SUBJECT MATTER:
US CL :

235/375; 345/329; 705/26, 27; 707/2, 3, 10, 100, 104, 200, 203, 506; 709/1, 201, 217, 218, 219, 235

フロントページの続き

(51)Int.Cl.[7]	識別記号	FI	テーマコード（参考）
G06F 17/30	419	G06F 17/30	419B

(81)指定国 EP(AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OA(BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG), AP(GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), EA(AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW